Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 9959304, 21 pages
https://doi.org/10.1155/2021/9959304

Research Article

Cooperative Edge Computing Task Offloading Strategy for Urban Internet of Things

Bo Wang[1,2,3] **and Mingchu Li**[1,3]

[1]*School of Software Technology, Dalian University of Technology, Dalian 116620, China*
[2]*School of Applied Technology, University of Science and Technology Liaoning, Anshan 114051, China*
[3]*Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian 116620, China*

Correspondence should be addressed to Bo Wang; wangboustl@126.com

Received 30 March 2021; Revised 30 August 2021; Accepted 4 September 2021; Published 7 October 2021

Academic Editor: Ning Wang

With the continuous progress of edge computing technology and the development of the Internet of Things technology, scenarios such as smart transportation, smart home, and smart medical care enable people to enjoy the smart era's convenience. Simultaneously, with the addition of many smart devices, a large number of tasks are submitted to the edge server, making the edge server unable to meet the needs of completing tasks submitted by the smart device. Besides, if the task is submitted to the remote cloud data center, it increases the user's additional delay and cost. Therefore, it is necessary to improve the task offloading strategy and resource allocation scheme to solve these problems. This paper first proposes a new task offloading mechanism and then proposes a two-stage Stackelberg game model to solve each participant's interaction problem in the task offloading mechanism and ensure the maximization of their respective interests. Finally, a theoretical analysis proves the equilibrium of the two-stage Stackelberg game. Experiments are used to prove the effectiveness of the proposed mechanism. Comparative experimental results show that the proposed model can achieve better results regarding delay and energy consumption.

1. Introduction

The urban Internet of things (IoT) plays a vital role in our daily life. It realizes smart city, urban brain, and other applications by processing the data generated by the intelligent devices deployed in the city. In the urban IoT scenario, the traditional processing method is cloud computing, because it can provide rich computing or storage resources to process much urban IoT data. However, when the task is submitted from an urban IoT device to a cloud computing data center, a large amount of data transmission seriously affects the processing performance, resulting in network congestion or high latency. Therefore, edge computing technology is used to offload tasks to edge servers or IoT devices. It is closer to end-users, to improve the system's performance in terms of service delay, QoS, and resource utilization. However, with the proliferation of smart devices in recent years, massive amounts of data are transmitted to edge servers or IoT devices. These bring a heavy burden to the communication bandwidth because the resources of the edge server or the IoT devices are limited. It inevitably leads to the phenomenon that multiple tasks compete to use the limited resources when providing services. A large number of resource contentions in edge servers undoubtedly lead to unbearable waiting delays, energy consumption, and degradation of end-user service quality. There are also many other edge servers or IoT devices in the urban IoT that may be idle or unused, which leads to the waste of resources. In addition, end-users obtain related services through payment. To obtain lower latency and higher satisfaction, users are willing to pay more, which inevitably causes users to spend more. Therefore, how can we offload the overloaded data to suitable processing facilities and provide high-quality services? How can we achieve low latency and high quality of user experience (QoE) for users with less expenditure and energy consumption? These have become an important issue.

In response to the above challenges, our goal is to design a method to utilize the unused or idle resources in the urban IoT. The task is offloaded to other servers or IoT devices to improve the efficiency of edge computing. In this paper, we call this pattern as "*cooperative edge computing* (CEC)." There are many types of research on collaborative edge computing. Ren et al. [1] propose to offload tasks to remote cloud computing centers for collaborative execution; it can use the system resources more effectively. Chen et al. [2] propose to offload tasks to multiple edge servers for collaborative execution. This method improves the performance of edge service providers. In [3], Device-to-Device (D2D) is utilized to offload tasks to other smart devices for collaborative execution. Various device resources are shared with other users through high-quality cellular connections. It can provide more and better services to attract more users. Jie et al. [4] propose an optimal resource allocation scheme for the IoT environment based on fog computing. To maximize resource utilization, the authors model the resource allocation problem as a two-stage Stackelberg game and propose three algorithms to achieve the Nash equilibrium and Stackelberg equilibrium. Although these methods all adopt the collaborative approach, they are fundamentally different from the model proposed in this paper. End-users perform the collaboration methods above by offloading tasks to remote cloud servers or other edge servers or IoT devices. The model proposed in this paper uses the edge scheduler to offload the overloaded tasks on the edge server to other idle edge servers or IoT devices for execution. The edge scheduler is also responsible for executing CEC task offloading strategies between other edge servers and IoT devices.

In the scenario with collaborative service providers, users hope to obtain more resources within their budget. Local edge service providers hope to obtain more revenue by providing services to more users. Collaborative service providers can get rewards by providing resources to local edge service providers. Users, local edge service providers, and collaborative service providers all pursue their interests. It is necessary to establish a feasible incentive mechanism to maximize interests for all participants and promote cooperation. Game theory is an effective tool to solve this problem.

This paper proposes a cooperative task offloading and resource allocation mechanism between edge servers and other edge servers or IoT devices. The proposed mechanism uses a two-stage Stackelberg game [5, 6] to solve matching tasks and resources in multiple rounds. Users give corresponding payments when the service quality meets their requirements. To increase the utilization rate of resources to save energy consumption and reduce costs, edge service providers match tasks with available resources in specific strategies according to users. According to particular strategies, the collaborative service provider matches the tasks submitted by the local edge service provider with the available resources. The two-stage game is based on the opponent's possible strategy, choosing the strategy to ensure that interests are maximized under its strategies to achieve the game equilibrium. The experimental results show that under the premise of ensuring high user satisfaction and not exceeding the user total budget, our proposed mechanism can obtain a near-optimal offloading strategy and is superior to traditional solutions in time delay and energy consumption.

The main contributions of this paper are as below.

(1) We establish delay, user satisfaction, cost, and energy consumption models and define formal task offloading and resource management problems in cooperative edge computing

(2) We propose a two-stage Stackelberg game model to solve the participant interactive problem in the task offloading mechanism and ensure the maximization of their respective interests

(3) We conduct a theoretical analysis of the game equilibrium of the two-stage Stackelberg game. Experiments are used to prove the effectiveness of the proposed mechanism

The remainder of this paper is organized as follows. In Section 2, we briefly review related works. In Section 3, we describe the system model and problem formulation. In Section 4, we propose a two-stage Stackelberg game model and analyze the model's game equilibrium. Section 5 shows the simulation results, and the conclusion is presented in Section 6.

2. Related Work

In recent years, researchers have made many studies in task offloading strategies and resource allocation for edge computing. Edge computing not only enriches cloud computing but also brings many challenges. Research on CEC has attracted wide attention.

Many studies deal with end-user offloading tasks collaboratively with multiple edge servers or base stations. Chen et al. [2] propose to offload tasks to multiple edge servers for collaborative execution. This method improves the performance of edge service providers. Fan and Ansari [7] design an application workload allocation scheme for the IoT based on edge computing. By determining the target cloudlet for each request of different IoT users, the amount of computing resources is allocated to each IoT user; IoT application requests' response time is minimized. Niu et al. [8] propose a workload allocation mechanism for the power IoT based on edge computing to minimize service delays. A workload optimization allocation model is established. Based on the optimization of computing resources in a single edge node, the optimal workload allocation between multiple edge nodes based on the delay is further realized. Hao et al. [9] propose a Smart-Edge-CoCaCo algorithm. To minimize the total delay and confirm the computational offloading decision, Smart-Edge-CoCaCo utilized the wireless communication model, the collaborative filter cache model in the edge cloud, and the joint optimization of the computational offloading model. Parwez and Rawat [10] propose a resource allocation method in an adaptive virtual wireless network with mobile edge computing. According to the demand of users, the authors study how to allocate mobile

virtual network operator resources to maximize revenue with satisfied service quality. Cooperative resource allocation is proposed to maximize the utilization ratio of mobile virtual network operators with desired QoS of user network speed. Zhang et al. [11] propose a collaborative task offloading scheme that considers the reuse of calculation results and minimizes energy consumption.

Many studies consider multiple influencing factors and handle tasks offloaded by end-users in a cloud-edge collaborative manner. Ren et al. [1] study the collaboration between cloud computing and edge computing, where tasks can be partially processed on edge nodes and cloud servers. The authors further transformed the communication and computing resource allocation problem into an equivalent convex optimization problem and obtained a closed-form resource allocation strategy. Jie et al. [4] propose an optimal resource allocation scheme for the IoT environment based on fog computing. To maximize resource utilization, the authors model the resource allocation problem as a two-stage Stackelberg game and propose three algorithms to achieve the Nash equilibrium and Stackelberg equilibrium. Guo et al. [12] propose a delay-based workload distribution scheme. It realizes the optimal workload distribution among local edge servers, neighboring edge servers, and remote clouds to achieve the minimum energy consumption of the IoT-edge-cloud system and a delay guarantee for arriving jobs. For the problem of cloud-mobile edge computing collaborative computing offloading, Guo and Liu [13] propose an approximate collaborative computing offloading scheme and a game theory cooperative computing offloading scheme as approximate solutions. Deng et al. [14] introduce an intermediate fog layer between mobile users and the cloud. An optimal workload distribution method between fog and cloud is proposed to solve the trade-off between power consumption and transmission delay in the fog-cloud computing system. The service delay is decreased with minimum power consumption. To minimize the average response time, mobile users offload their application workloads to geographically dispersed cloudlets. He et al. [15] provide a collaborative computing offloading example based on energy consumption, computing power, variable transmission power, and remaining battery power. To handle delay-sensitive tasks effectively, they designed an iterative search algorithm for a collaborative computing offloading scheme to minimize task offloading overhead. To minimize energy consumption under the conditions of ensuring service completion time, Liu et al. [16] propose an energy-saving collaborative task computing offloading algorithm based on semideterministic relaxation and random mapping methods. Wu et al. [17] propose an edge-cloud collaborative multitask computing offloading model. The latency and energy costs are considered. By converting the model solution to a search solution in limited strategy space, it is solved by a nonlinear exponential inertia weighted particle swarm optimization algorithm. By dynamically adjusting the inertia weight, the premature convergence defect of the standard particle swarm algorithm can be compensated. The optimal local solution can be effectively avoided. Li et al. [18] propose a computing offloading mechanism based on the Stackelberg game to analyze the interaction between multiple edge clouds and numerous industrial IoT devices. The payment cost is considered. The author also formulates the revenue function by considering the social interaction information from potential industrial IoT devices.

Some scholars use cloud-edge-end collaboration to handle tasks offloaded from end-users. Hossain et al. [19] propose a collaboration model. It can dynamically offload computing tasks' execution to the SBS-MEC server and mobile devices or remote cloud. In [20], one task can offload subtasks. The subtask can be offloaded according to the characteristics of the edge server (such as transmission distance and central processing unit capacity). The authors propose a low-complexity adaptive offloading scheme based on the Hungarian algorithm using a multi-subtask multiserver model for new applications that require real-time information work.

Some scholars use D2D collaboration to implement collaborative edge computing. Ciobanu et al. [21] propose a computational offloading solution that can improve user QoE, reduce application and service developers' cost, and reduce battery consumption. In [3], the D2D is utilized to offload tasks to other smart devices for collaborative execution. Various device resources are shared with other users through high-quality cellular connections. It can provide more and better services to attract more users.

Some scholars implement collaborative edge computing in an edge-to-end cooperative manner. He et al. [22] propose an effective method to find the best solution to optimize the system allocation time, transmit power and CPU on each device, and maximize the amount of data that two users can process in a given time frame. Kim et al. [23] propose a new concept of IoT-assisted edge computing, which provides edge services by integrating idle resources in IoT devices and offloading tasks to nearby IoT devices.

Although these methods all adopt the collaborative approach, they are fundamentally different from the model proposed in this paper. End-users perform the collaboration methods above by offloading tasks to remote cloud servers or other edge servers or IoT devices. The model proposed in this paper uses the edge scheduler to offload the overloaded tasks on the edge server to other idle edge servers or IoT devices for execution. The edge scheduler is also responsible for executing CEC task offloading strategies between other edge servers and IoT devices. In addition, due to the user time tolerance and the user total budget constraints, relevant research does not consider the impact of user costs and user satisfaction. Therefore, the main research content of this paper is to integrate idle trusted devices within a short distance from the end-user, assist the local edge service provider to complete the end-user offloading tasks, and jointly consider the impact of the end-user cost and QoE factors on the offloading strategy from the user view.

3. System Model and Problem Formulation

As shown in Figure 1, the entire system consists of end-users, edge service providers, edge scheduler, and

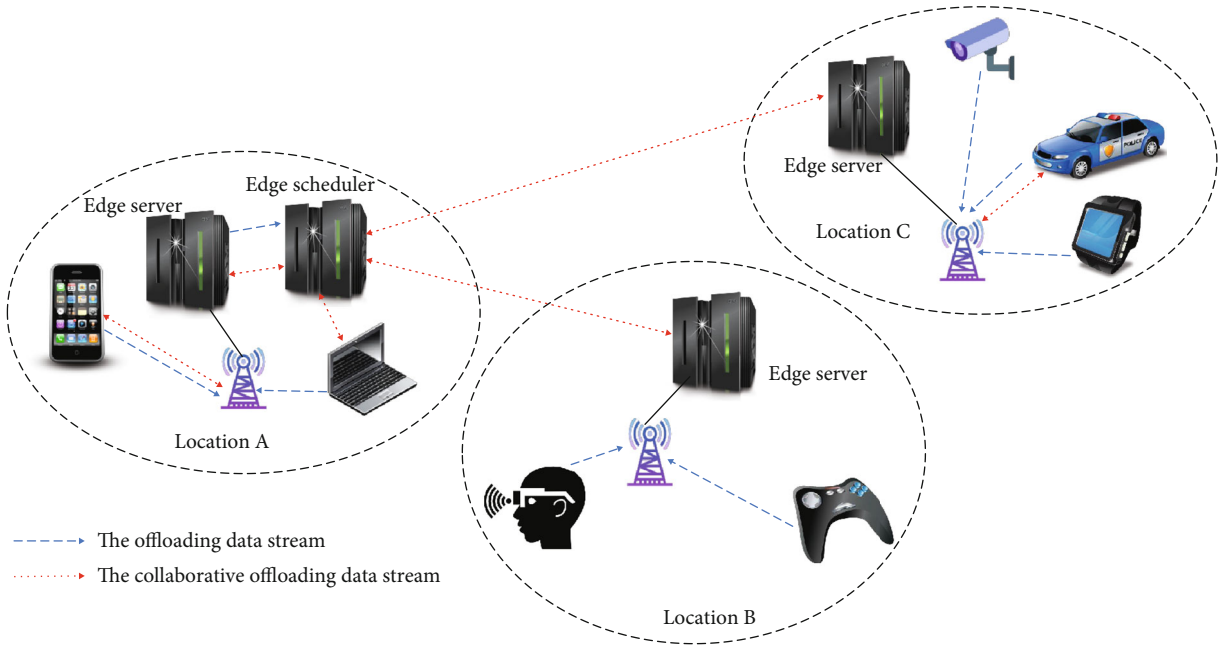


Figure 1: Multiedge resource cooperative offloading mechanism system structure diagram.

collaborative service providers. Smart devices include mobile smart terminals, IoT devices, notebooks, smart medical devices, and the Internet of Vehicles. Here, $EU = \{eu_1, eu_2, \cdots, eu_N\}$ represents the set of all devices in the smart device, where N represents the number of devices. The edge service provider is composed of base stations with edge servers distributed across different locations. The edge service provider can handle tasks submitted by related smart devices in their area and tasks migrated by other edge devices if resources are available. The set $ESP = \{esp_1, esp_2, \cdots, esp_M\}$ represents the collection of all edge servers in the edge service provider, where M represents the number of edge servers in the edge service provider. The set $R = \{R_1, R_2, R_3, \cdots, R_N\}$ is the set of resources required by all smart device users. The collaborative service provider includes other edge servers in the edge service provider and other smart devices with idle resources. The collaborative service provider can help the local edge device complete the smart device tasks as needed. For example, for the edge server at location A in Figure 1, when the resources of the edge server cannot meet user requirements, the task can be offloaded to the edge server at location B or C where the resources are available for execution. We call the edge server at location A local edge server and the edge server at location B or C collaborative edge servers. Besides, at position A in Figure 1, the smartphone's task can also be offloaded to the laptop for execution while waiting in the waiting queue of the local edge device. We call the laptop at this time as a collaborative smart device.

In the urban IoT scenario, the edge scheduler runs the task offloading decision mechanism to allocate task offloading strategies. Tasks are reasonably scheduled to different edge servers or edge devices by the edge scheduler, which can realize the collaborative processing of tasks between edge servers and edge devices. In this paper, cloud data centers are generally far from edge servers and devices. The offloading task to remote cloud data centers takes a long time and consumes more energy. Therefore, we do not consider the implementation of task offloading to remote cloud data centers. The mathematical symbols are summarized in Table 1.

3.1. Delay Model. As shown in Figure 1, the end-user offloads the task to the service device of the edge service provider. The end-user offloads the task to the base station wirelessly, and then, the base station transmits the task to the edge server through the optical fiber. This part of the delay can be expressed by

$$T_o = \sum_{i=1}^{\Gamma} \left(\frac{B_i^o}{C_i^o} + \frac{B_i^o}{F_i^o} \right). \tag{1}$$

Here, B_i^o represents the size of the offloading task i, C_i^o represents the transmission rate of the uploading in the wireless method, and F_i^o represents the transmission rate of the uploading using the fiber method. Γ represents the number of offloading tasks.

When a large number of tasks are waiting on the local edge server, the edge scheduler offloads tasks to other collaborative edge servers and other collaborative smart devices to reduce the waiting time and increase the completion rate of tasks.

When the local edge service provider has no resources to use, some tasks can be offloaded to the server of the cooperative edge service provider for execution. In this case, the delay of this part includes task transmission time $T_{i,ne}^o$, task execution time T_i^{ne}, and return time after task completion $T_{i,ne}^d$. Since the parallel mode is adopted, the delay is

Table 1: Symbol summary.

Symbol	Definition
T	Total delay
E	Total energy consumption
C^{co}	The cost of end-user
p_i^c	The price of CPU for the edge server
p_i^s	The price of storage for the edge server
p_i^m	The price of memory for the edge server
C^{esp}	The cost of edge service provider
p_i^{ne}	The price of other collaborative edge servers executing various types of resources
p_i^{ns}	The price of other collaborative smart devices executing various types of resources
Q_i	The satisfaction score of user i
bid_i	The bidding price of the smart device
p_i	The edge service provider's resource price

calculated according to the longest time required to complete the offloading task:

$$T^{ne} = \sum_{i=1}^{\kappa}\left(T_{i,ne}^{o} + T_i^{ne} + T_{i,ne}^{d}\right) = \sum_{i=1}^{\kappa}\left(\frac{B_{i,ne}^{o}}{C_o^{ne}} + \frac{B_{i,ne}^{o}}{\text{Cap}_{ne}} + \frac{B_{i,ne}^{d}}{C_d^{ne}}\right). \tag{2}$$

Here, κ represents the number of tasks that are offloaded to the server of the collaboration service provider for execution. $B_{i,ne}^{o}$ represents the size of the task i that is offloaded to the neighboring collaborative edge server, and C_o^{ne} represents the local edge server's transmission rate to the collaborative edge server. Cap^{ne} represents other collaborative edge devices' computing power. $B_{i,ne}^{d}$ represents the size of the backhaul data of the offloading task i from the neighboring collaborative edge server, and C_d^{ne} represents the backhaul data transmission rate from the collaborative edge server to the local edge server.

When the local edge service provider has no resources to use, some tasks can be offloaded to the edge device of the collaborative edge service provider for execution. In this case, this part of the delay includes task transmission time $T_{i,ns}^{o}$, task execution time T_i^{ns}, and return time after the task is completed $T_{i,ns}^{d}$. Since it is carried out in parallel, the delay is calculated according to the longest time required to complete the offloading task:

$$T^{ns} = \sum_{i=1}^{l}\left(T_{i,ns}^{o} + T_i^{ns} + T_{i,ns}^{d}\right) = \sum_{i=1}^{l}\left(\frac{B_{i,ns}^{o}}{C_o^{ns}} + \frac{B_{i,ns}^{o}}{\text{Cap}^{ns}} + \frac{B_{i,ns}^{d}}{C_d^{ns}}\right). \tag{3}$$

Here, l represents the number of tasks performed offloaded to the smart device of the collaboration service provider. $B_{i,ns}^{o}$ represents the offloaded task i's size to the neighboring collaborative smart device, and C_o^{ns} represents the local edge server's transmission rate to the collaborative smart device. Cap^{ns} represents the computing capabilities of other collaborative smart devices. $B_{i,ns}^{d}$ represents the size of the back-transfer data of the offloaded task i from the neighboring collaborative smart device, and C_d^{ns} represents the backhaul data transmission rate from the collaborative smart device to the local edge server.

When the tasks are completed, the edge service provider returns the result to the end-user. The delay of this part can be expressed by

$$T^{d} = \sum_{i=1}^{\Gamma}\left(\frac{B_i^d}{F_i^d} + \frac{B_i^d}{C_i^d}\right). \tag{4}$$

Here, B_i^d represents the size of the backhaul data of the offloading task i, F_i^d represents the transmission rate of the backhaul using the fiber method, and C_i^d represents the transmission rate of the backhaul in the wireless method.

In summary, the total time delay to complete the task can be expressed by

$$T = T^{o} + T^{ne} + T^{ns} + T^{d}. \tag{5}$$

3.2. Energy Consumption Model. According to the Shannon formula, the transmission rate of uploading data and the transmission rate of backhauling data are shown in equations (6) and (7):

$$C_i^o = W_o \times \log_2^{\left(1+P_i^o \times G/P_n\right)}. \tag{6}$$

Here, W_o represents the bandwidth of the wireless transmission channel from the smart device to the local edge server, P_i^o represents the smart device's transmit power during the process of wirelessly uploading data, G represents the gain of wireless transmission channel from the smart device to the local edge server, and P_n represents the Gaussian

white noise power. $P_i^o \times G/P_n$ is the signal-to-noise ratio:

$$C_i^d = W_d \times \log_2^{\left(1+P_i^d \times G/P_n\right)}. \tag{7}$$

Here, W_d represents the bandwidth of the wireless backhauling data transmission channel from the local edge server to the smart device and P_i^d represents the transmit power of the local edge server in the process of backhauling data in the wireless link mode. According to the research in [24], $G = 103.8 + 20.9 \times \log_{10}^{(\text{dist})}$, where dist represents the distance between the smart device and the local edge server.

According to equations (6) and (7), we then obtain the energy consumption from the smart device to the local edge server and the energy consumption of the backhauling process after the task is completed, as below:

$$E^o = \sum_{i=1}^{\Gamma}\left(P_i^o \times \frac{B_i^o}{C_i^o} + P_f^o \times \frac{B_i^o}{F_i^o}\right), \tag{8}$$

$$E^d = \sum_{i=1}^{\Gamma}\left(P_i^d \times \frac{B_i^d}{C_i^d} + P_f^d \times \frac{B_i^d}{F_i^d}\right). \tag{9}$$

Here, P_f^o represents the upload power of optical fiber transmission and P_f^d represents the backhaul power of optical fiber transmission.

When the edge service provider has no resources to use, some tasks can be offloaded to the servers of other collaborative edge service providers or smart devices for execution. This part of the energy consumption includes the energy consumption of tasks transferred to the collaboration server and transfer to the collaboration energy consumption of edge devices. In this case, the energy consumption can be expressed by equations (10) and (11), respectively:

$$E_{ne}^o = \sum_{i=1}^{\kappa}\left(P_{ne}^o \times \frac{B_{i,ne}^o}{C_o^{ne}}\right), \tag{10}$$

$$E_{ns}^o = \sum_{i=1}^{l}\left(P_{ns}^o \times \frac{B_{i,ns}^o}{C_o^{ns}}\right). \tag{11}$$

Here, P_{ne}^o represents the power offloaded from the local edge device to the cooperative edge server and P_{ns}^o represents the power offloaded from the local edge device to the collaborative smart device.

When executed on the server of the collaborative edge service provider, the energy consumption of this part includes the energy consumption of task execution and the energy consumption of the return after the task is completed. In this case, the energy consumption can be expressed by

$$E^{ne} = \sum_{i=1}^{\kappa}\left(P_{ne}^{im} \times \frac{B_{i,ne}^o}{\text{Cap}^{ne}} + P_{ne}^d \times \frac{B_{i,ne}^d}{C_d^{ne}}\right). \tag{12}$$

Here, P_{ne}^{im} represents the power performed by the cooperative edge server and P_{ne}^d represents the backhaul power after the completion of the cooperative edge device. $B_{i,ne}^o$ represents the size of task i offloaded to the adjacent collaborative edge server, and Cap^{ne} represents the computing power of the collaborative edge server. $B_{i,ne}^d$ represents the size of the returned data after task i is offloaded to the adjacent collaborative edge server for execution, and C_d^{ne} represents the backhaul data transmission rate from the collaborative edge server to the local edge server.

When executed on smart devices of collaborative edge service providers, this part of the energy consumption includes task execution energy consumption and backhaul energy consumption after the task is completed. In this case, the energy consumption can be expressed by

$$E^{ns} = \sum_{i=1}^{l}\left(P_{ns}^{im} \times \frac{B_{i,ns}^o}{\text{Cap}^{ns}} + P_{ns}^d \times \frac{B_{i,ns}^d}{C_d^{ns}}\right). \tag{13}$$

Here, P_{ns}^{im} represents the power executed by the collaborative smart device and P_{ns}^d represents the backhaul power after the completion of the collaborative smart device. $B_{i,ns}^o$ represents the size of the task i offloaded to the adjacent collaborative smart device, and Cap^{ns} represents the computing power of collaborative smart devices. $B_{i,ns}^d$ represents the size of the returned data after task i is offloaded to the neighboring collaborative smart device for execution, and C_d^{ns} represents the transmission rate of backhaul data from the collaborative smart device to the local edge server.

In summary, the total energy consumption to complete the task can be expressed by

$$E = E^o + E_{ne}^d + E^{ne} + E_{ns}^d + E^{ns} + E^d. \tag{14}$$

3.3. Cost Model. Each edge service provider completes various tasks of different types submitted from end-users by providing relevant resources. Each edge service provider incurs expenses when performing tasks. The cost function of the edge service provider can be defined as below:

$$C^{\text{esp}} = \sum_{i=1}^{\Gamma}\left(p_i^c \times B_i^o \times \Omega_i\right) + \sum_{i=1}^{\Gamma}\left(p_i^s \times B_i^o \times \xi_i\right) + \sum_{i=1}^{\Gamma}\left(p_i^m \times B_i^o \times \zeta_i\right), \tag{15}$$

where Γ represents the number of offloaded tasks; Ω_i, ξ_i, and ζ_i are coefficients; p_i^c represents the price of CPU for the edge server; p_i^s represents the price of storage for the edge server; and p_i^m represents the price of memory for the edge server.

When a large number of tasks are waiting to be executed in the edge server, the waiting tasks can be offloaded to other collaborative edge servers and another collaborative edge device. In this case, the cost function of the cooperative service provider can be defined as shown in

$$C^{co} = \sum_{i=1}^{\kappa}\left(p_i^{ne} \times B_{i,ne}^o \times \Upsilon_i\right) + \sum_{i=1}^{l}\left(p_i^{ns} \times B_{i,ns}^o \times \Psi_i\right). \tag{16}$$

Here, κ represents the number of tasks offloaded to other collaborative edge servers and l represents the number of tasks offloaded to other collaborative edge devices. $\Upsilon_i = 1$ represents that task i is executed on the collaborative edge server; otherwise, $\Upsilon_i = 0$, and $\Psi_i = 1$ represents that task i is executed on the collaborative smart device; otherwise, $\Psi_i = 0$. p_i^{ne} represents the price of other collaborative edge servers executing various types of resources, and p_i^{ns} represents the price of other collaborative edge devices executing various types of resources.

3.4. User Satisfaction Model. In this paper, we introduce the user satisfaction model to evaluate the satisfaction degree of users. Based on the literature [25, 26], the satisfaction function can be defined as below:

$$Q_i = \beta \times \log_{\alpha}^{(\psi_i(1-p_i/\mathrm{bid}_i))}. \tag{17}$$

Here, α and β represent the correlation coefficient. ψ_i represents the priority of the task i submitted by the user. The larger the value of Q_i, the more satisfied the user i is. On the contrary, the more dissatisfied the user i is.

3.5. Problem Formulation. In urban IoT, many smart devices submit tasks to various edge service providers and realize their demand with payment. Due to the distance among each edge service provider device, the energy consumption of smart devices, and the quality of network communication, many tasks cannot reach the system simultaneously. We assume that each task can use multiple different resources, such as CPU, memory, and storage. Edge service providers have a large number of edge servers that can handle various types of tasks. When the edge server resources are not available, the task can be offloaded to other collaborative edge devices and collaborative smart devices collaboratively to complete tasks more quickly to ensure user satisfaction. CEC resource allocation is performed by the edge scheduler.

In the above description, it can be seen that the collaborative service provider has idle resources and obtains corresponding benefits by selling idle resources. In this paper, the utility of the collaborative service provider is expressed as

$$U^{co} = \theta_c(R^{\mathrm{esp}} - C^{co}) - \theta_e E^{co} - \theta_t T^{co}. \tag{18}$$

Here, R^{esp} represents that the cooperative service provider obtains revenue from the edge service provider, as shown in equation (19). The coefficients θ_e, θ_t, and θ_c, respectively, represent the weight coefficients of energy consumption, delay, and cost in the utility function of the cooperative service provider, with values ranging from 0 to 1:

$$R^{\mathrm{esp}} = \sum_{i=1}^{\Gamma} \left(\mathrm{pay}_i^{\mathrm{esp}} \times B_i^o\right). \tag{19}$$

C^{co} represents the cost of the collaborative service provider to perform the offloading task of the edge service provider, as shown in equation (16).

E^{co} represents the energy consumption of the collaborative service provider's offloading task of the edge service provider. Energy consumption includes the energy consumption of the execution and the return result, as shown in

$$E^{co} = \sum_{i=1}^{\Gamma} (E_i^{ns} + E_i^{ne}). \tag{20}$$

T^{co} represents the execution delay of the collaborative service provider, as shown in

$$T^{co} = \sum_{i=1}^{\Gamma} (T_i^{ns} + T_i^{ne}). \tag{21}$$

Therefore, the goal of collaborative service providers is to maximize their utility, and the utility optimization problem of collaborative edge service providers can be defined as shown in

$$\max_{j \in M} U_j^{co} \tag{22}$$

$$\text{s.t.} \begin{cases} \mathrm{C1}: \mathrm{pay}_i^{\mathrm{esp}} \geq p_i^{ns}, \\ \mathrm{C2}: \mathrm{pay}_i^{\mathrm{esp}} \geq p_i^{ne}, \\ \mathrm{C3}: \mathrm{Cap}_{ns}^{\min} \leq \mathrm{Cap}_i^{ns} \leq \mathrm{Cap}_{ns}^{\max}, \\ \mathrm{C4}: \mathrm{Cap}_{ne}^{\min} \leq \mathrm{Cap}_i^{ne} \leq \mathrm{Cap}_{ne}^{\max}, \\ \mathrm{C5}: 0 \leq \sum_{j=1}^{M} \sum_{i=1}^{\Gamma} R_{i,j} \leq R^{\max}, \\ \mathrm{C6}: \sum_{j=1}^{M} \sum_{i=1}^{\Gamma} E_{i,j}^{ns} \leq E_{\mathrm{battery}}. \end{cases} \tag{23}$$

Here, Γ represents the number of offloading tasks and M represents the number of edge service providers. T_{tolerate} represents the user's maximum tolerable waiting time, and E_{Battery} represents the maximum available battery of the collaborative edge service provider. $R^{\max}$ represents the number of maximum available resources. $\mathrm{Cap}_{ns}^{\min}$ and $\mathrm{Cap}_{ns}^{\max}$ represent the minimum and maximum processing capacity of the smart device of the collaborative service provider, respectively; $\mathrm{Cap}_{ne}^{\min}$ and $\mathrm{Cap}_{ne}^{\max}$ represent the minimum and maximum processing capacity of the server of the collaborative service provider, respectively. C1 and C2 represent that the fee paid by the edge service provider must be greater than or equal to the cost of the collaborative service provider to ensure that the collaborative service provider can provide services normally. $\mathrm{pay}_i^{\mathrm{esp}}$ represents the fees paid by the local edge service provider. C3 and C4 ensure that the offloaded tasks do not exceed the processing capacity of the collaborative service provider. C5 indicates that the resources allocated by the collaboration service provider are within the permitted range of available resources. C6 ensures that the energy consumption does not exceed the allowable range of battery power when performing tasks

on the smart devices allocated to the collaboration service provider.

The edge service provider purchases the resources of the collaborative service provider and still provides paid services for one or more end-users under the condition of limited resources. This paper expresses the utility function of the edge service provider as shown in

$$U^{\text{esp}} = \upsilon_c(R^{\text{eu}} - C^{\text{esp}}) - \upsilon_e E^{\text{esp}} - \upsilon_t T^{\text{esp}}. \tag{24}$$

Here, R^{eu} represents the service fee paid by the end-user to complete the task, as shown in equation (25). υ_e, υ_t, and υ_c represent the weight of energy consumption, delay, and cost of the edge service provider:

$$R^{\text{eu}} = \sum_{i=1}^{\Gamma} (\text{bid}_i \times B_i^o). \tag{25}$$

C^{esp} represents the cost of the edge service provider to perform the offloading task, as shown in equation (15).

E^{esp} represents the energy consumption generated by the edge service provider offloading the task to the cooperative service provider, as shown in

$$E^{\text{esp}} = \sum_{i=1}^{\Gamma} \left(E_{i,ns}^t + E_{i,ne}^t\right). \tag{26}$$

T^{esp} represents the transmission delay of the edge service provider, as shown in

$$T^{\text{esp}} = \sum_{i=1}^{\Gamma} \left(T_{i,ns}^o + T_i^{ns} + T_{i,ns}^d + T_{i,ne}^o + T_i^{ne} + T_{i,ne}^d\right). \tag{27}$$

Therefore, the goal of the edge service provider is to maximize its utility, and the utility optimization problem of the edge service provider can be defined as shown in

$$\max_{j \in N} \quad U_j^{\text{esp}} \tag{28}$$

$$\text{s.t.} \quad \text{C1} : R^{\text{eu}} \geq R^{\text{esp}}. \tag{29}$$

Here, N represents the number of end-users, and C1 represents that the revenue must be greater than or equal to the system expenditure to ensure the normal operation of the system.

End-users submit tasks that need to be processed to edge service providers. Because they need to pay relevant fees, they can obtain resources to perform the tasks. This paper defines the utility function of the end-user as shown in

$$U^{\text{eu}} = Q - \rho_c \times R^{\text{eu}} - \rho_e E^{\text{eu}} - \rho_t T^{\text{eu}}. \tag{30}$$

Here, ρ_e, ρ_t, and ρ_c represent the weights of energy consumption, delay, and cost of the end-user, respectively.

E^{eu} represents the energy consumption when the end-user offloads tasks to the edge service provider, as shown in

$$E^{\text{eu}} = \sum_{i=1}^{\Gamma} E_i^o. \tag{31}$$

T^{eu} represents the time delay for the end-user to complete the task, as shown in

$$T^{\text{eu}} = \sum_{i=1}^{\Gamma} T_i. \tag{32}$$

Therefore, the goal of each end-user is to minimize the payment, energy consumption, and time delay and maximize user satisfaction. The utility optimization problem of the end-user can be defined as shown in

$$\max_{j \in N} \quad U_j^{\text{eu}} \tag{33}$$

$$\text{s.t.} \begin{cases} \text{C1} : R^{\text{eu}} \leq \text{Budget}, \\ \text{C2} : \text{bid} \geq p, \\ \text{C3} : \sum_{i=1}^{\Gamma} B_i^o \leq B^{\text{total}}, \\ \text{C4} : B_i^o \geq 0, \\ \text{C5} : E^{\text{eu}} \leq E_{\text{battery}}, \\ \text{C6} : T^{\text{eu}} < T_{\text{tolerate}}. \end{cases} \tag{34}$$

Here, Γ represents the number of offloading tasks and N represents the number of end-users. T_{tolerate} represents the maximum tolerable waiting time of the user. E_{Battery} represents the maximum available battery power of the end-user, and B^{total} represents the upper limit of the executable task size. C1 ensures that the total cost of the task does not exceed the total budget. C2 guarantees that the edge service provider only provides the corresponding service when the user's payment is greater than or equal to the minimal price of the edge service provider. If the cost value paid by the user is less than the minimum price of the edge service provider, the edge service provider refuses to provide the service. C3 and C4 ensure that the offloading tasks can be executed. C5 represents that the energy consumption of transmission must be within the allowable range of battery power. C6 ensures that the delay does not exceed the maximum tolerable waiting time of the user.

Through the above analysis, the task offloaded by the end-user can be regarded as multiple objects, and the resources of the collaborative service provider can be regarded as boxes, and then, the problem can be transformed into the multiobjective maximum packing problem. It can be proved that such problems are NP-hard problems. This type of problem is more difficult to solve. Next, the game theory is used to find an approximate solution to the problem, and the validity of the solution is proved. This

paper uses the entropy weight method [27] to calculate the value of each parameter.

4. Proposed Mechanism

Different from the traditional edge computing task offloading mechanism, this paper takes other collaborative edge devices and other smart devices into consideration in the design of the task offloading mechanism. In this paper, the proposed task offloading mechanism has three types of participants who affect each other: smart devices and edge service providers and collaborative service providers. Their respective behavior strategies determine their ultimate benefits. For example, when a collaborative service provider provides services to an edge service provider, if the fee paid by the edge service provider to the collaborative service provider is too low, the resources of collaborative service providers for providing collaborative services to the edge service provider are correspondingly reduced. In this case, the delay for edge service providers to provide services to smart devices will increase, affecting the satisfaction and benefits of customers.

Conversely, when the edge service provider pays too much to the collaborative service provider without increasing user fees, this leads to a significant reduction to the benefits of the edge service provider. When the edge service provider is overpriced for providing services to smart devices, the smart devices will submit fewer tasks or not to the edge service provider. It will reduce the revenue of the edge service providers. On the contrary, when the price of the edge service provider is too low, a large number of tasks will be submitted to the edge service provider, which will increase the expenses of the edge service provider or reduce the actual income. It will increase the waiting time of smart devices and affect the quality of service. We can see interest relationships among smart devices, edge service providers, and collaborative service providers from the above analysis. Therefore, we propose a task offloading mechanism based on a two-stage Stackelberg game model. By calculating the equilibrium solution of the Stackelberg game, we can obtain the optimal resource allocation scheme. This paper assumes that the cooperative service providers are secure and reliable.

4.1. Game Model. In this paper, we propose a two-stage Stackelberg game model. The specific description of this model is given as below.

Stage 1: the collaborative service provider submits the number of its remaining resources and the corresponding price to the edge scheduler. The edge scheduler, as the leader of the game, submits the relevant price strategy to the collaborative service provider according to the actual condition. As a follower of the game, the cooperative service provider decides its resource allocation strategy according to the leader's strategy. Algorithm 1 is used to solve the resource allocation problem of the edge scheduler.

Stage 2: we regard the interaction between the edge service provider and each smart device as a repeated Stackelberg game with multiple participants. On the one hand, as long-term participants, edge service providers obtain profits by processing various tasks of smart devices. On the other hand, as short-term participants, smart devices complete the tasks by paying fees to edge service providers. Algorithm 2 is used to solve the problem of the task offloading strategy of users. In each round of the Stackelberg game, the edge service provider, as the leader, first chooses its bidding strategy. Then, the smart devices, as followers, can decide their bidding strategy based on the edge service provider devices and their prediction for the next round of the game. The edge scheduler is responsible for executing the bidding strategy of end-users and edge service providers and allocating resources according to the final price. When there are multiple edge service devices in the vicinity of a smart device, each smart device will choose single or multiple edge service providers to submit tasks for execution based on the price of each edge service provider. The bidding strategy of the edge service provider is shown in Algorithm 3. The processing diagram of this game model is shown in Figure 2.

4.2. Game Equilibrium Analysis. According to the proposed game model, end-users, edge service providers, and collaborative service providers can maximize their respective benefits. In this section, we will analyze the equilibrium of the two-stage Stackelberg game model.

Theorem 1. *The two-stage Stackelberg game proposed in this paper has a Nash equilibrium among end-users, edge service providers, and collaborative service providers.*

The detailed certification process is given in the appendix.

5. Numerical Results and Discussion

5.1. Experimental Parameter Setting. In order to simulate the collaborative edge computing scenario of the urban IoT, we set up an experimental environment. The specific configuration is as follows: Within the range of 1×1 (km^2), there are one cloud server, one edge server, two collaborative edge servers, and twenty collaborative smart devices. Collaborative smart devices are all smartphones. It can be extended to more edge devices and more edge servers, with similar results.

The task size varies from 0 to 9 G. The bandwidth between the edge and the cloud is 1 Gbps. The bandwidth between the edge and the collaborative edge is 54 Mbps. The bandwidth between the edge and the smart device is 40 Mbps. The CPU processing capacity of the cloud data center is 10 G cycles/s, and the CPU processing capacity of the edge server is 6 G cycles/s. For the edge server, its idle power is 135 W. Its peak power is 495 W. The number of cores is 22. For the cloud computing server, the idle power is 150 W. The peak power is 750 W, and the number of cores is 64 [28]. The prices of different resources of the edge service provider are as below: 3 for CPU resources, 0.1 for storage resources, and 0.05 for memory resources. The prices of different resources for cloud resource providers are 24 for CPU resources, 0.82 for storage resources, and 0.67 for

Input:
B_i, location information for collaborative service provider, the number of the collaboration service provider, the set $\mathscr{C}'$ is a subset of the cooperative offloading service provider strategy set $\mathscr{C}$
Output:
The optimal strategy set $\mathcal{S}$ of the edge scheduler
1: A weighted directed graph G can be generated according to the location of the collaborative service provider
2: While true do
3: Calculate the delay, energy consumption, and cost of offloading tasks
4: Using dynamic programming to solve the equilibrium solution $\mathcal{S}$ of the subgame on the cooperative offloading service provider strategy set $\mathscr{C}'$
5: Solve the optimal corresponding strategy $\mathbb{C}$ of the cooperative offloading service provider to the subgame equilibrium solution $\mathcal{S}$
6: If $\mathbb{C} \in \mathscr{C}'$ then
7: Return $\mathcal{S}$
8: Else
9: $\mathscr{C}' \leftarrow \mathscr{C}' \cup \{\mathbb{C}\}$
10: End If

Algorithm 1: Edge scheduler resource allocation algorithm.

Input:
Number of end-users N, Resource requirements of each end-user R_i, Each end-user pays the resource price bid_i, The size of the end-user offloading task B_i^o, End-user's budget $Budget$
Output:
End-user task offloading strategy
1: For $i = 1$; $i <= N$ do
2: if $bid_i \times B_i^o <= Budget$ then
3: if the resources of local edge service provider j meet the needs of end-user i then
4: The task of end-user i is offloaded to the local edge service provider j for execution;
5: beak;
6: endif
7: else if the resources of local edge service provider j do not meet the needs of end user i then
8: Calculate the size of the remaining unfinished tasks of the end-user i on the local edge service provider j;
9: Calculate the remaining budget of the end-user i;
10: endif
11: endif
12: endfor

Algorithm 2: End-user task offloading algorithm.

memory resources. The power of task upload is 1.3 W, and the power of task return is 1.2 W.

The available resources of the system are key indicators that will affect the performance of the system. In the experiment, we use the percentage of unoccupied resources to indicate the availability of resources. The percentage of unoccupied resources is set to 10%, 30%, 50%, 70%, and 90%, respectively. When the percentage of unoccupied is 10%, the system is extremely busy. The number of devices that can be serviced by the collaboration service provider is minimal. The system is idle when the percentage of unoccupied is 90%. The collaboration service provider can provide more services.

We compare three offloading modes with our model. The details are as follows:

(i) *Edge Execution*. Task offloaded to edge server for execution

(ii) *Edge and Cloud Collaborative Execution*. Task offloaded to edge and cloud for collaborative execution

(iii) *Cloud Execution*. Task offloaded to cloud server for execution

(iv) *Our Model*. Task offloading to other collaborative edge servers on the adjacent edges and (or) other smart devices for execution

5.2. Experimental Results and Analysis

5.2.1. Experiment 1: The Performance of the Proposed Mechanism. When the task size submitted by users varies from 1 to 9 G and the fee paid by the local edge service provider to the collaborative service provider varies from 5 to 30, Figure 3 shows the optimal utility determined by the local edge service provider under the Stackelberg equilibrium condition. We can observe that with the increase of

Input:
Number of edge service providers M,Number of tasks submitted by end-users Γ,end-user bids $\{bid_1, bid_2,\cdots,bid_m\}$, Bids of edge service providers $\{p_1, p_2,\cdots,p_N\}$, Resource requirements of each end-user R, the available resources RES, Small changes in the price of local edge service providers Δp_k
Output:
The optimal bidding strategy set S of the local edge service provider
1: For $i = 0$; $i <= M$ do
2: For $j = 0$; $j <= \Gamma$ do
3: if $bid_j < p_i$ and $R > RES$ then
4: Denial of service;
5: else
6: Submit the task to the waiting queue;
7: endif
8: endfor
9: endfor
10: Sort the tasks in the waiting queue in descending order of the end user's bid
11: For $i = 1$; $i <= M$ do
12: For $k = 1$; $\mathbf{k} <= L$ do
13: Calculate the energy consumption, cost, and time delay of the task
14: if $U_i^{esp}(p_k+\Delta p_k) \le U_i^{esp}(p_k)$ and $U_i^{esp}(p_k-\Delta p_k) \le U_i^{esp}(p_k)$ then
15: $S \longleftarrow S \cup \{p_k\}$
16: else if $U_i^{esp}(p_k+\Delta p_k) > U_i^{esp}(p_k)$ and $U_i^{esp}(p_k-\Delta p_k) \le U_i^{esp}(p_k)$ then
17: $S \longleftarrow S \cup \{p_k+\Delta p_k\}$
18: else if $U_i^{esp}(p_k+\Delta p_k) \le U_i^{esp}(p_k)$ and $U_i^{esp}(p_k-\Delta p_k) > U_i^{esp}(p_k)$ then
19: $S \longleftarrow S \cup \{p_k-\Delta p_k\}$
20: else if $U_i^{esp}(p_k+\Delta p_k) > U_i^{esp}(p_k)$ and $U_i^{esp}(p_k-\Delta p_k) > U_i^{esp}(p_k)$ then
21: $S \longleftarrow S \cup \max\{p_k-\Delta p_k, p_k+\Delta p_k\}$
22: endif
23: endfor
24: endfor
25: Return S

Algorithm 3: Bidding strategies of the edge service provider.

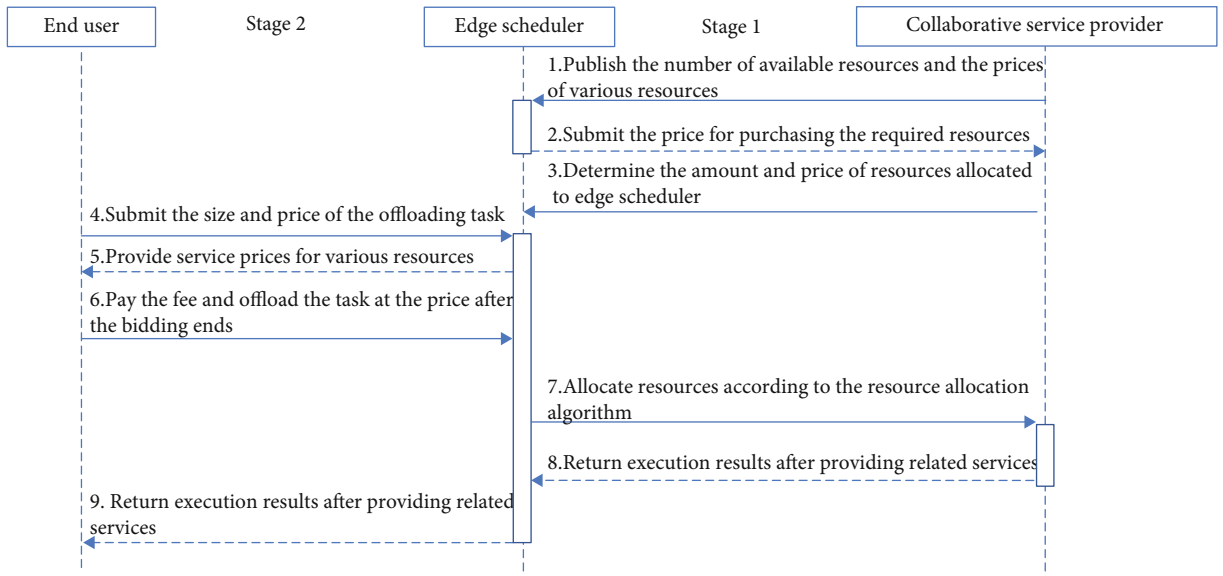


Figure 2: The sequence diagram of the game model.

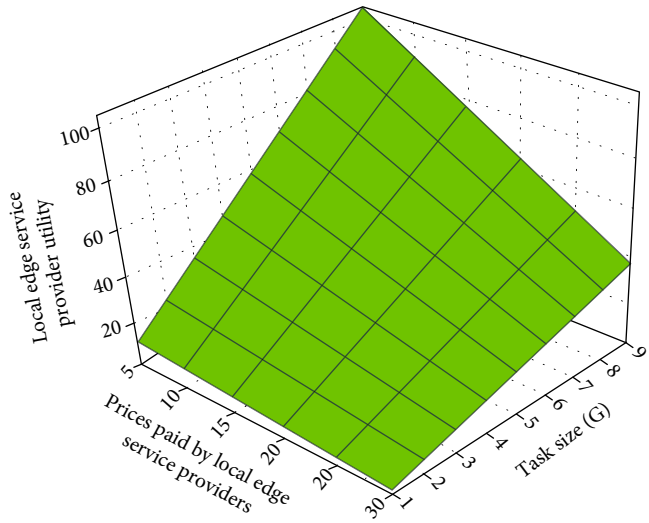


Figure 3: The utility of local edge service providers under different task sizes.

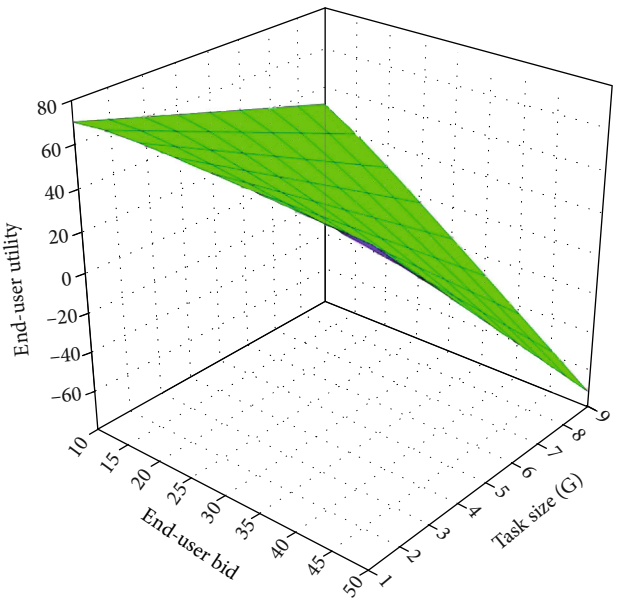


Figure 4: End-user utility under different task sizes.

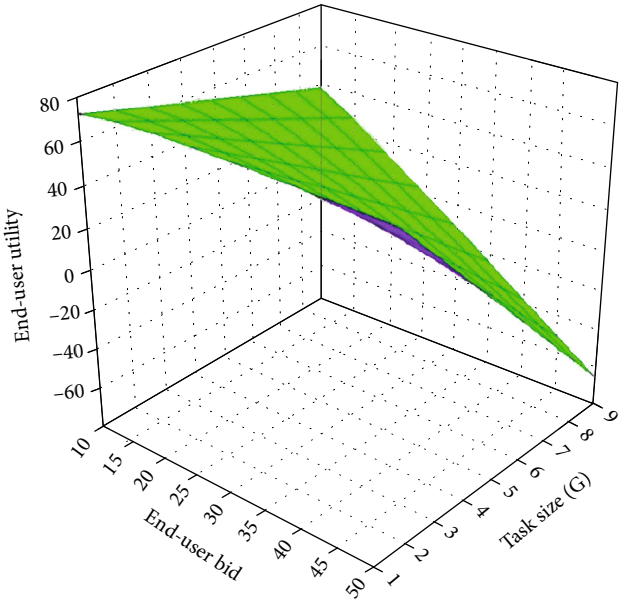
the number of tasks, the utility value of the local edge service increases correspondingly. To motivate the collaborative service provider to complete the tasks of the local edge provider, the utility value of the local edge service provider is reduced when the payment to the collaborative service provider is increased. The user's bid is fixed at 50 in this experiment.

When the task size varies from 1 to 9 G and the fee paid by the user varies from 10 to 50, Figure 4 shows the optimal utility determined by the end-user under the Stackelberg equilibrium condition. We can observe that with the increase of the number of tasks submitted by users, the time and energy consumption of the task execution increase correspondingly. To ensure user satisfaction, the utility value of end-users decreases. With the increase of the fee paid by users, users' expenses increase. It will directly lead to the decrease of utility value.

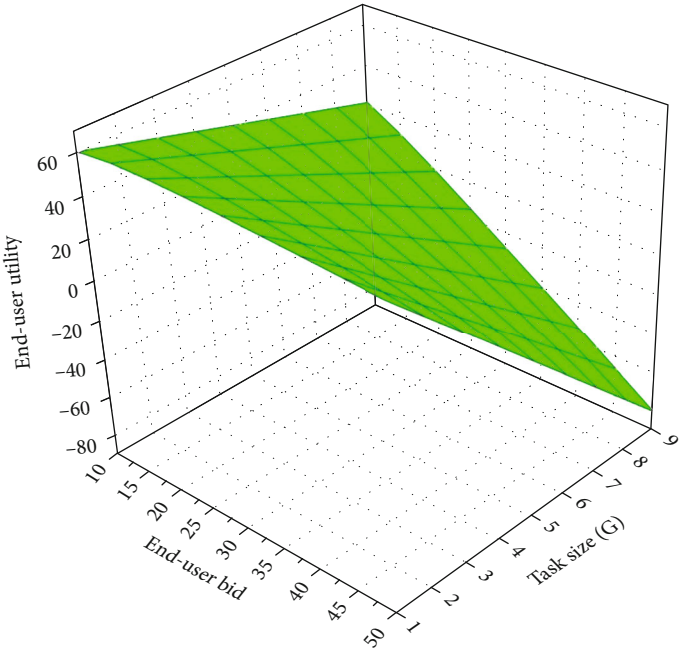
When the task size varies from 1 to 9 G and the fee paid by the user varies from 10 to 50, Figure 5 shows the end-user's optimal utility under the Stackelberg equilibrium condition. The priority of the task ranges from 0.1 to 1. The higher the priority value of the task submitted by the end-user, the more priority the task is executed. Due to space limitations, only $\psi = 0.5$ and $\psi = 0.75$ are shown, and other cases are similar to them. Figure 5(a) shows the result with $\psi = 0.75$, and Figure 5(b) shows the result with $\psi = 0.5$. It can be observed that with the increase of the number of tasks, the task priority is higher, and the user's utility value is greater. High-priority tasks are executed first. Moreover, high-priority tasks can get far more resources than low-priority tasks.

Figure 6 shows the optimal utility determined by the collaborative service provider under Stackelberg equilibrium when the collaborative service provider's task size varies from 1 to 9 G and the payment of the local edge service provider varies from 5 to 30. It can be observed that with the increase of the number of tasks processed by the collaborative service provider, tasks will be assigned to different collaborative service providers to execute in parallel; the corresponding execution time and energy consumption will be less than that of the sequential execution. Correspondingly, the utility of collaborative service providers will become greater and greater. When the number of tasks performed does not change, with the increase of payment of the local edge service provider, the revenue of the collaborative service provider and the utility value increase correspondingly.

When the proportion of unoccupied resources in the system varies from 10% to 90%, and the payment of the local edge service provider to the collaborative service provider varies from 5 to 30, Figure 7 shows the optimal utility of the local edge service provider under the Stackelberg



(a) The utility of the end-user when $\psi = 0.75$



(b) The utility of the end-user when $\psi = 0.5$

Figure 5: The utility of end-users under different priorities and different task sizes.

equilibrium condition. The task size is fixed at 9 G in this experiment. We can observe that with the increase of the unoccupied resources of the system, the system has more resources to process tasks submitted by end-users. Meanwhile, the utility value of the local edge service increases. To encourage collaborative service providers to complete the tasks of local edge providers, the utility value of local edge service providers decreases with the increase of fees paid to collaborative service providers. It can be explained as follows: with the increase of the fee provided by the local edge service providers, the profit value of local edge service providers decreases. We can observe that when the resource utilization rate is below 50%, the utility of the local edge service provider does not change much. When the system resource utilization rate is above 50%, the utility of the local edge service provider is greatly affected.

Figure 8 shows the end-user's optimal utility under the Stackelberg equilibrium condition when the proportion of unoccupied resources in the system varies from 10% to 90%, and the fee of the end-user varies from 10 to 50. It can be observed that with the increase of unoccupied resources of the system, the system can use more resources to handle the tasks submitted by end-users. Moreover, the execution time and energy consumption of users are

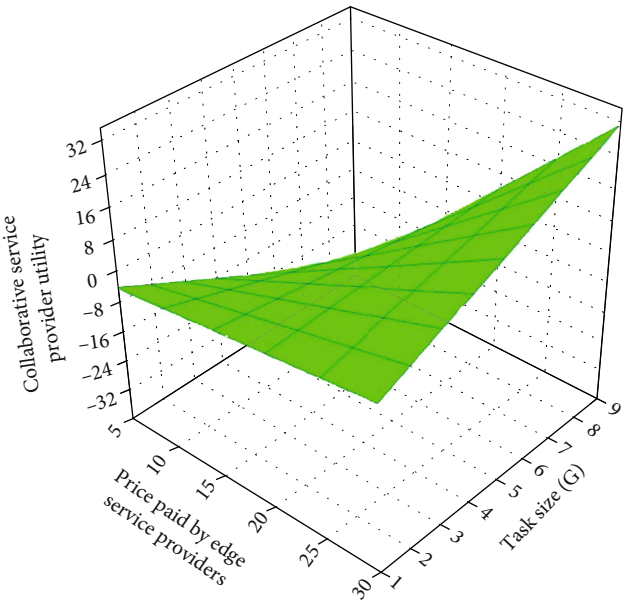


Figure 6: The utility of collaborative service provider in different task sizes.

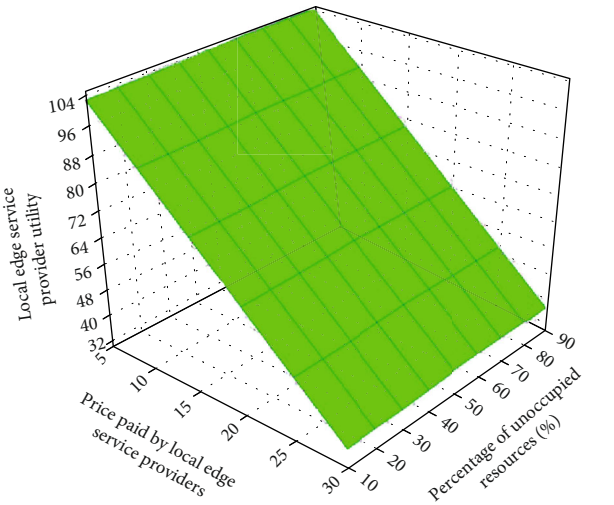


Figure 7: The utility of local edge service providers under different unoccupied resources.

reduced. Therefore, the utility value of end-users is reduced to ensure user satisfaction. To encourage the local edge service provider to complete the task, the end-user increases their payment under the condition of constant system utilization. When the utility value is less than 20, with the increase of the fee, the end-user's change is more significant. It indicates that the user expense has a greater impact on the utility value in this case. When the utility value is greater than 20 and less than 50, the change of the end-user's utility value is small. It indicates that the impact of the user expense on the utility value is small in this case.

Figure 9 shows the optimal utility of the user under the Stackelberg equilibrium condition when the proportion of unoccupied resources in the system varies from 10% to

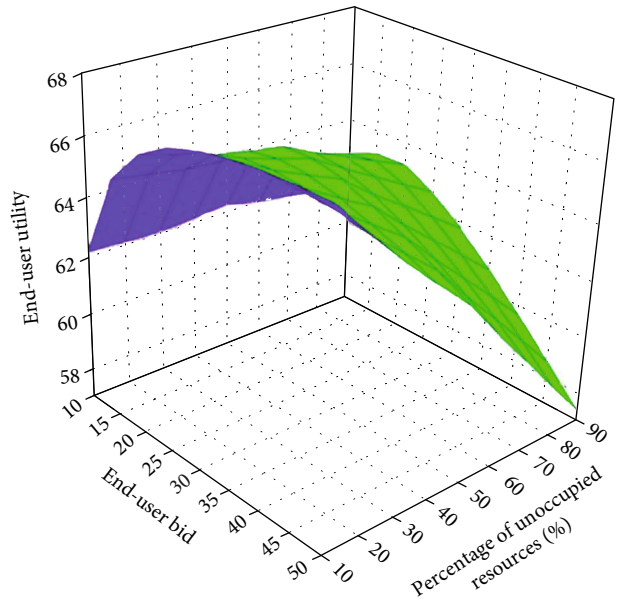
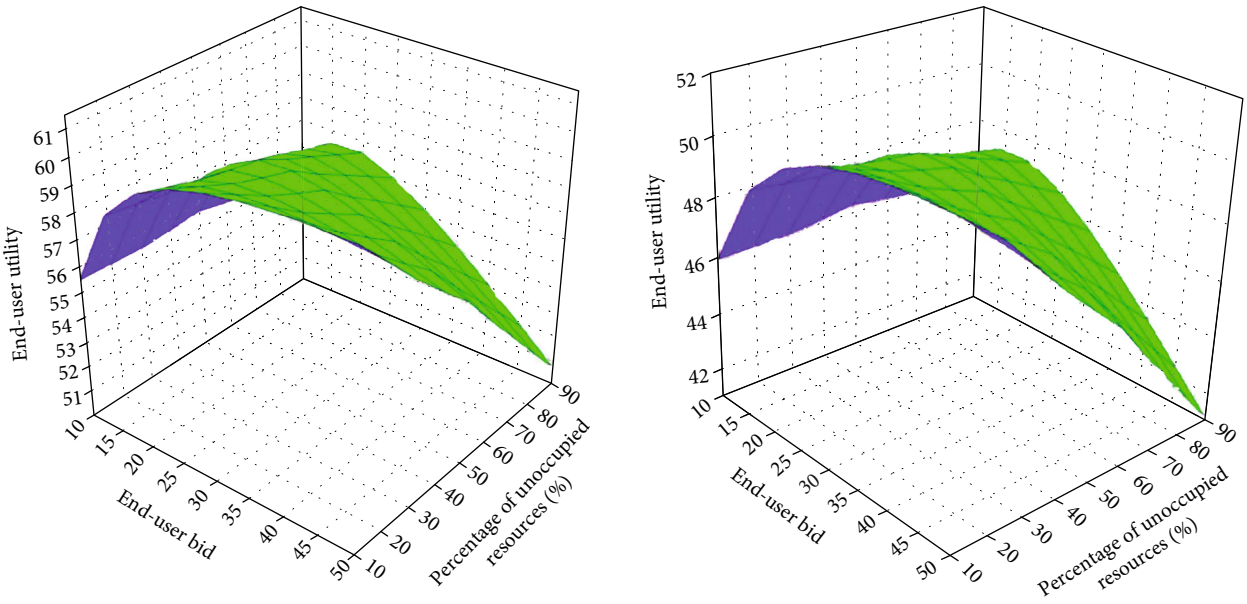


Figure 8: End-user utility under different resource utilization.

90%, and the user expense varies from 10 to 50. The priority of the task ranges from 0.1 to 1. The higher the priority value of the task submitted by the end-user, the more priority the task is executed. Due to space limitations, only $\psi = 0.5$ and $\psi = 0.75$ are shown, and other cases are similar to them. Figure 9(a) shows the result with $\psi = 0.75$. Figure 9(b) shows the result with $\psi = 0.5$. It can be observed that with the increase of the unoccupied resources of the system, the system can use more resources to process tasks. Meanwhile, the execution time and energy consumption of users are reduced. Therefore, the utility value of the user is reduced. Besides, with the increase of the priority of the task, the utility value of the user increases. The reason is that with the increase of the task priority, the waiting delay of the task decreases. More resources can be obtained for high-priority tasks.

Figure 10 shows the optimal utility determined by the collaborative service provider under Stackelberg equilibrium conditions when the proportion of unoccupied resources in the system varies from 10% to 90% and the payment of the local edge service provider varies from 5 to 30. It can be observed that with the increase of the unused resources of the system, the collaborative service provider has more resources to process the tasks submitted by the end-user. Therefore, the utility value of the collaborative service provider increases. It can be observed that when the remaining available resources are in the range of 10% to 30%, the improvement of the utility for the collaboration service provider is slower. The reason is that available resources are limited and the demand for a large number of tasks cannot be met in this case. With the increase of the number of available resources in the range of 30% to 70%, it can be found that the speed of processing tasks increases due to the increase of available resources. The execution time of the



(a) The utility of the end-user when $\psi = 0.75$

(b) The utility of the end-user when $\psi = 0.5$

Figure 9: End-user utility under different priorities and different resource utilization rates.

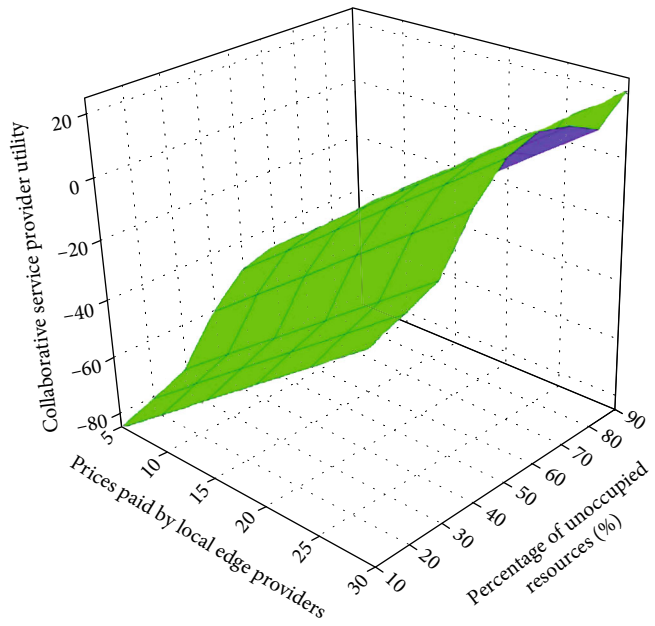


Figure 10: The utility of collaborative service providers in the case of different unoccupied resources.

task decreases. The revenue of the collaboration service provider increases significantly. When a large number of devices are in idle condition, the task execution time decreases slowly. The growth of the utility value for the collaborative service provider slows down.

5.2.2. Experiment 2: The Performance Comparison of Task Offloading Mechanisms with Different Task Sizes. We assign the task size for task offloading mechanisms from 0 to 9 G to evaluate how it affects the energy consumption and delay in the system. Figures 11 and 12, respectively, show the comparison results of the energy consumption and delay of computing offloading in different collaborative service modes under different task scales. The unused rate of resources is fixed at 90% in this experiment.

Figure 11 shows the comparison results of the energy consumption under different task sizes. It can be seen from Figure 11 that with the increase of the task number, energy consumption increases. By comparing the model proposed in this paper with cloud execution, edge execution, and

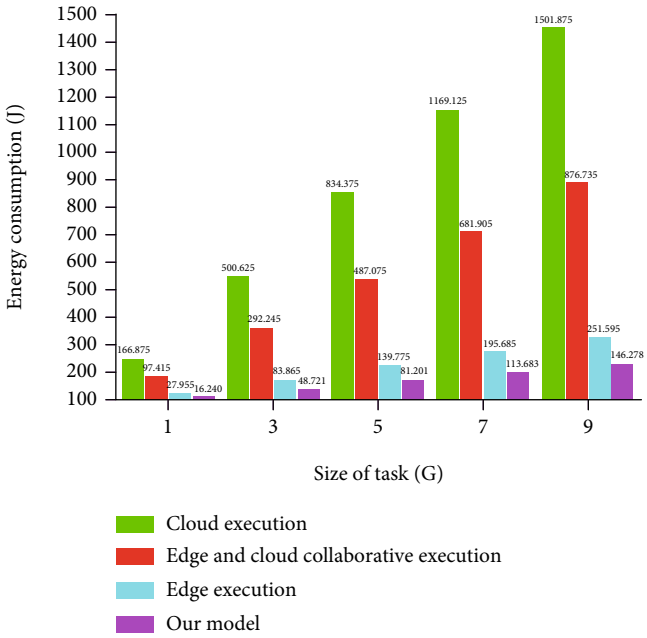


Figure 11: Comparison of energy consumption for different task sizes.

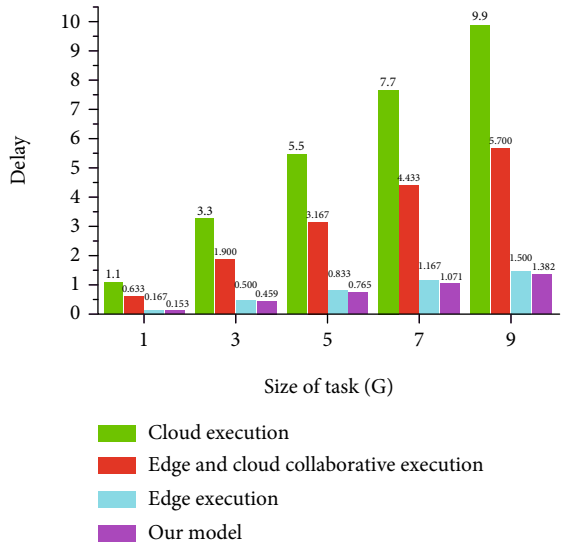


Figure 12: Comparison of delay for different task sizes.

edge-cloud collaborative execution, we observe that the energy consumption of the model proposed in this paper is less than that of the other models. When the task size is small (less than 1 G), the difference in energy consumption between the proposed model and the reference model is small. However, when the task size is large (larger than 9), the difference in energy consumption between the proposed model and the reference model is huge. All offloading to the cloud is 1501.875 J, while the energy consumption of the collaboration method proposed in this paper is 146.278 J. In terms of energy consumption, it can be seen that the model proposed in this paper has good results. As the number of tasks increases, the performance improved significantly.

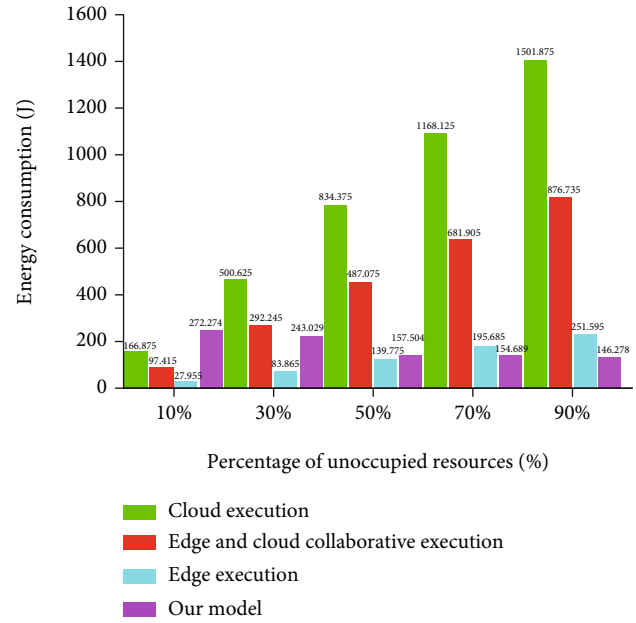


Figure 13: Comparison of energy consumption for different resource occupancies.

Figure 12 shows the comparison results of the delay under different task sizes. It can be seen from Figure 12 that with the increase of the number of tasks, the delay for tasks to be executed also increases. Compared to cloud execution, edge execution, and edge-cloud collaborative execution methods, the proposed model has better delay performance. When the task load is small (1 G), the difference of the delay for different methods is not significant, the longest value is 1.1 s, and the lowest is 0.153 s. However, when the task volume is large (9 G), the delay for all offloading tasks is 9.9 s, while the delay caused by the collaboration method proposed in this paper is 1.382 s. In terms of delay, it can be seen that the model proposed in this paper has good results. As the number of tasks increases, the model proposed in this paper also has better performance.

5.2.3. Experiment 3: The Performance Comparison of Task Offloading Mechanisms with Different Resource Usage. We assign the resource usage for task offloading mechanisms from 10% to 90% to evaluate how it affects the energy consumption and delay in the system. Figures 13 and 14, respectively, show the comparison results of the energy consumption and delay of computing offloading in different collaborative service modes under different resource usage. The task size is fixed at 9 G in this experiment.

Figure 13 shows the comparison results of the energy consumption under different resource usage. It can be seen from Figure 13 that with the increase of the percentage of unoccupied resources for collaboration service providers, more tasks will be executed. The energy consumption will increase correspondingly. By comparing the model proposed in this paper with cloud execution, edge execution, and edge

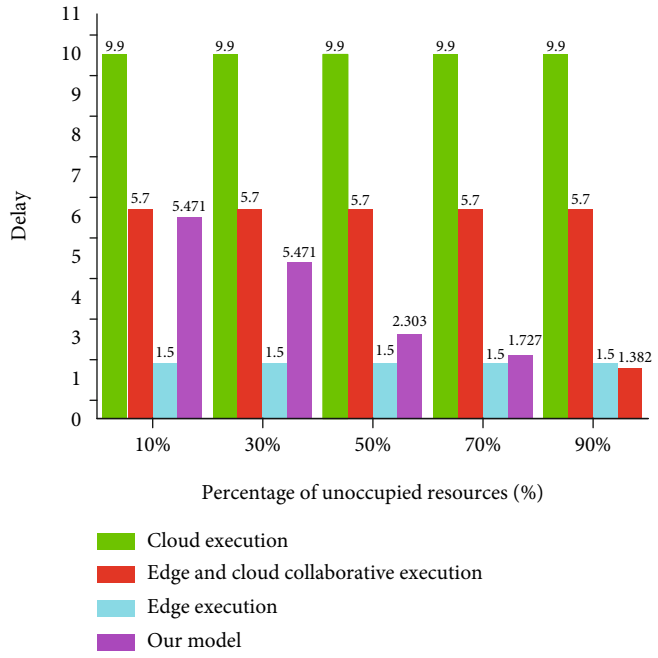
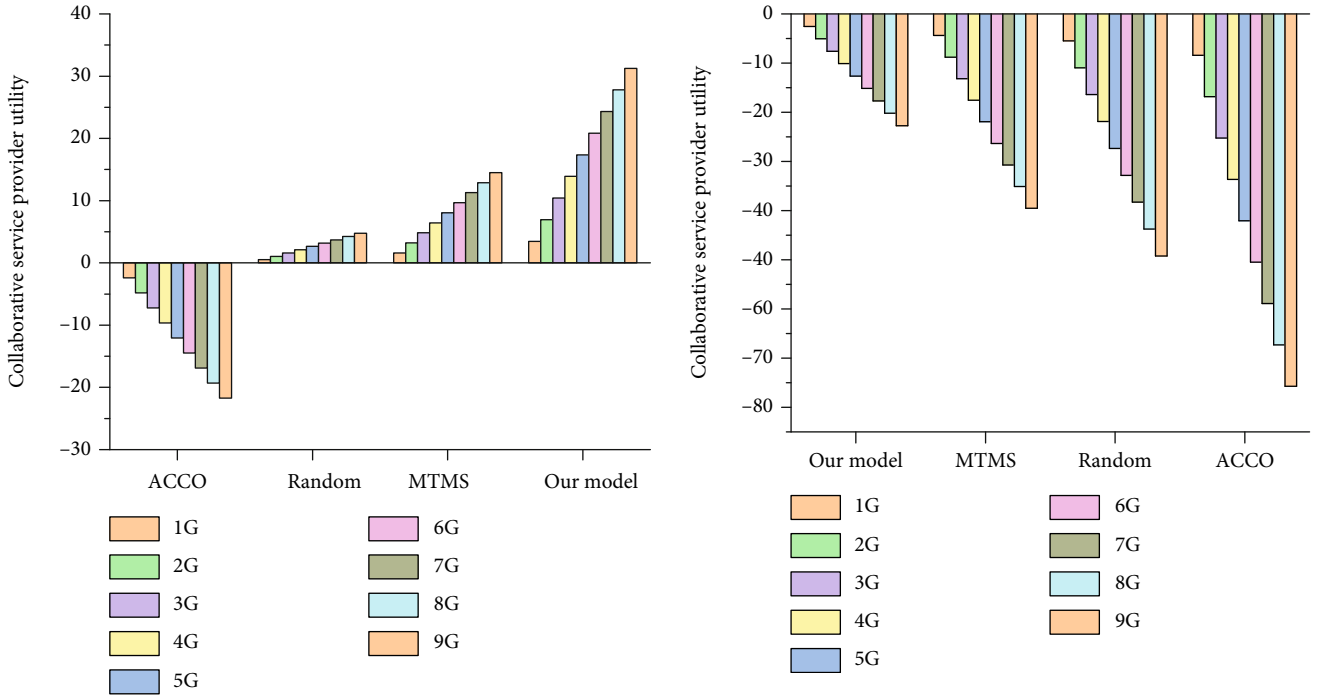


Figure 14: Comparison of delay for different resource occupancies of collaborative service providers.

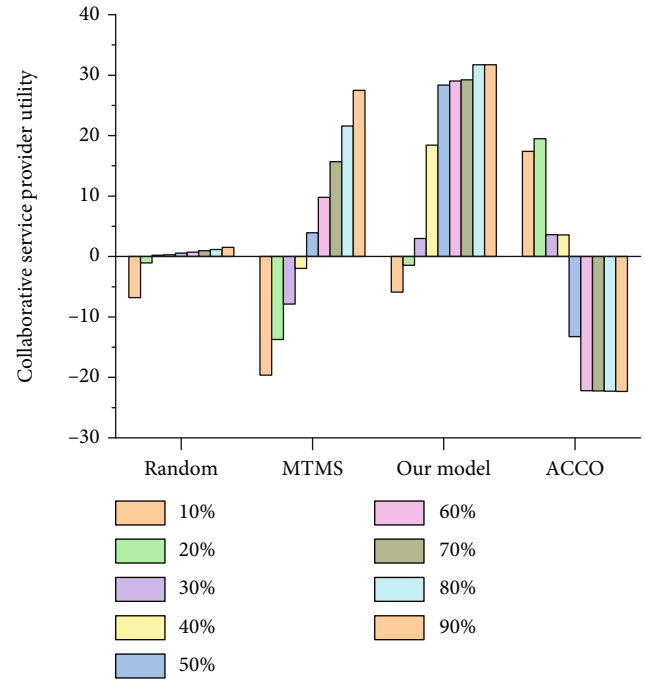


(a) The utility of the collaborative service provider utility when $pay^{esp} = 30$ (b) The utility of the collaborative service provider utility when $pay^{esp} = 10$

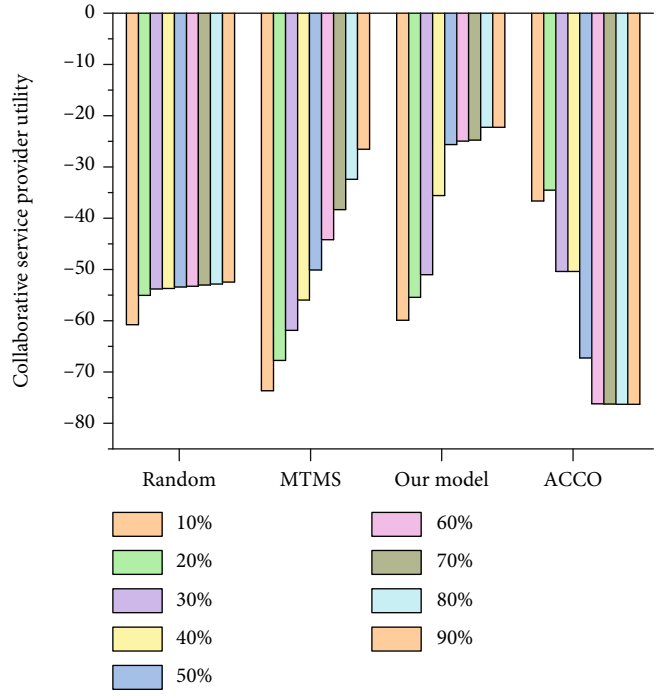
Figure 15: Comparison of collaborative service provider utility for different task sizes.

and collaborative cloud execution, it is found that the energy consumption of the model proposed in this paper is less than that of the other models when the unoccupied percentage of resources of the collaborative service provider is more than 50%. The energy consumption of the model proposed in this paper is higher than edge execution when the resource utilization rate is less than 50%. This result shows that when the collaborative service provider does not have free resources to complete all offloading tasks, many tasks are offloaded to the collaborative service providers and generate more energy than that executed at the edge.

Figure 14 shows that with the increase of the number of unoccupied resources for the collaboration service provider, more free resources are obtained, and more tasks are



(a) The utility of the collaborative service provider utility when $pay^{esp} = 30$ and task size = 9



(b) The utility of the collaborative service provider utility when $pay^{esp} = 10$ and task size = 9

Figure 16: Comparison of collaborative service provider utility for different resource occupancies.

executed. The latency of the model proposed in this paper is less than that of the other models. Compared to cloud execution, edge execution, and edge-cloud collaborative execution, when the percentage of unoccupied resources of the collaborative service provider is less than 70%, the latency of the proposed model is higher than that of the edge execution. It means that the collaborative service provider does not have free resources to complete all the offloading tasks. The offloading of many tasks to the collaborative service provider will not be processed. Therefore, there will be a larger delay compared to the execution at the edge.

5.2.4. Experiment 4: Comparison of the Proposed Mechanism, ACCO, MTMS, and Random. Experiment 1 shows that the proposed mechanism is affected by the change of its variables. Experiments 2 and 3 merely show that the

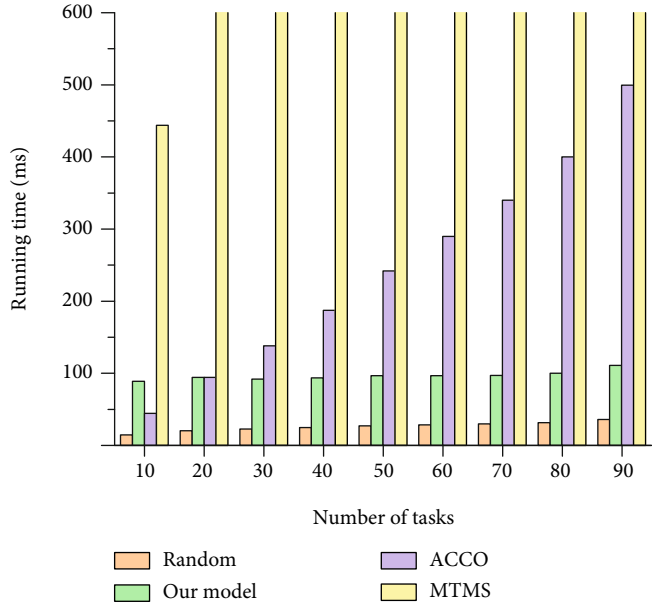


Figure 17: Comparison of the algorithm running time for different numbers of tasks.

performance comparisons of task offloading mechanisms are affected by the task size and resource usage. To prove the good performance of the proposed mechanism, we make a comparison of it with ACCO presented in [13], MTMS presented in [20], and the random allocation mechanism.

Figure 15(a) shows the utility of collaborative service providers for different task sizes when the edge service provider pays 30. It can be seen from the experimental results that the performance of the proposed scheme is better than the other three mechanisms because the edge scheduler in this mechanism obtains resources and services from edge servers and edge devices. The difference is that MTMS first divides the task into subtasks and then allocates the subtasks from the edge server where other resources are idle. When other edge servers are far away or resources are limited, there will be transmission or waiting-related delay, cost, and energy consumption. ACCO offloads the task to the remote cloud. As the size of the task increases, the user's energy consumption and time delay will increase, which will inevitably lead to a decrease in the utility value while the payment fee remains unchanged. Figure 15(b) shows the utility of collaborative service providers for different task sizes when the edge service provider pays 10. It can be seen from the experimental results that as the size of the task increases, when the edge service provider's pay is low, the utility function value of each mechanism has declined, and the decline rate of the scheme proposed in this paper is lower than that of the other three mechanisms. This shows that the delay and energy consumption of the proposed scheme in this paper are smaller than other schemes. However, the utility value of each mechanism is negative, indicating that the payment of edge service providers cannot meet the requirements of encouraging cooperative service providers to provide services.

Figure 16(a) shows the utility function value for different available resource percentages of collaborative service providers when the edge service provider pays 30. The experimental results show that with the increase of the scale of available resources, when the pay of edge service providers is high, only the utility value of the ACCO mechanism decreases, and other utility function values increase. This shows that when the available resources are only 10% to 30%, a large number of tasks are waiting to be executed, resulting in increased delay and energy consumption. However, the ACCO mechanism offloads tasks to the remote cloud. Because the processing capacity of the cloud data center is higher than that of other devices, the processing time and energy consumption are lower than other mechanisms. Therefore, the ACCO mechanism will be superior to other mechanisms. However, when the idle resources are higher than 30%, the energy consumption and cost of execution in the cloud are higher than other mechanisms due to factors such as distance, energy consumption, and cost. Therefore, other utility value increases, and the utility value of the ACCO mechanism decreases. Figure 16(b) shows the utility function value for different available resource percentages of collaborative service providers when the edge service provider pays 10. It can be seen from the experimental results that as the scale of available resources increases when the edge service provider's pay is low, the changing trend of the utility function value of each mechanism is similar to the result in Figure 16(a). However, the utility value of each mechanism is negative, indicating that the payment of edge service providers cannot meet the requirements of encouraging cooperative service providers to provide services.

5.2.5. Experiment 5: Running Time Comparison. Figure 17 shows the execution time of each mechanism under the different number of tasks. It can be seen from the figure that as the number of tasks increases, the execution time of each mechanism increases. Random runs the shortest, and MTMS runs the longest. This is because available resources

are randomly selected in the random mechanism, and the task will be offloaded as long as the execution conditions are met, while other methods need to consider the influencing factors such as energy consumption, delay, and cost and other influencing factors to make a comprehensive decision, so the execution time of other mechanisms is higher than random. However, it can be seen from previous experiments that although the execution time of the model proposed in this paper is not as good as the random mechanism, it is better than the random mechanism in other performances.

6. Conclusion

In order to reduce the time and energy consumption of task processing, tasks on edge servers with limited resources are offloaded to collaborative edge servers and edge devices for execution. Based on the collaboration task offloading mechanism, this paper proposed a two-stage Stackelberg game model to solve the interactive problem of the participants in the task offloading mechanism. The new proposal ensures the maximization of interests for all participants. Experiments and simulations verify the effectiveness of our method.

Appendix

Lemma 1. *The set of cooperative service providers' strategies can maximize their profits, and the optimal strategy is unique.*

Proof. According to equations (22) and (23), the first-order and second-order partial derivatives of the size of the offloading tasks from the local edge service provider to the collaborative service provider can be written as equations (35) and (36). Because the first derivative is greater than zero, the second derivative is equal to zero. Therefore, the function is incremental, there will be a maximum point, and there will only be one. The conclusion is established:

$$\frac{\partial U_i^{co}}{\partial B_i^o} = \theta_c\left(\text{pay}_i^{\text{esp}} - \sum_{i=1}^{l}\left(p_i^{ns} \times \Psi_i\right)\right) - \theta_e \sum_{i=1}^{l}\left(\frac{p_{ns}^{im}}{\text{Cap}^{ns}}\right) - \theta_t \sum_{i=1}^{l}\left(\frac{1}{C_o^{ns}} + \frac{1}{\text{Cap}^{ns}}\right), \tag{35}$$

$$\frac{\partial^2 U_i^{co}}{\partial B_i^{o2}} = 0. \tag{36}$$

□ □

Lemma 2. *The set of end-user's strategies can maximize their profits, and the optimal strategy is unique.*

Proof. According to equations (33) and (34), the first-order and second-order partial derivatives of the bid of the end-user are shown in

$$\frac{\partial U_i^{\text{eu}}}{\partial \text{bid}_i} = \sum_{i=1}^{\Gamma}\left(\frac{\beta \times p_i}{\text{bid}_i(\text{bid}_i - p_i) \times \ln \alpha} - \rho_c \times B_i^o\right), \tag{37}$$

$$\frac{\partial^2 U_i^{eu}}{\partial \text{bid}_i^2} = \sum_{i=1}^{\Gamma}\left(\frac{-(2 \times \text{bid}_i - p_i) \times \beta \times p_i}{\ln \alpha \times \left(\text{bid}_i^2 - p_i \times \text{bid}_i\right)^2}\right). \tag{38}$$

The second derivative is less than 0. Therefore, U_i^{eu} is a convex function. Since the function is increasing and convex, there is only one maximum point in the function. Therefore, the conclusion is established.□ □

Lemma 3. *The edge service provider's strategy set can maximize the benefits, and the optimal strategy is unique.*

Proof. In stage 2, the income of edge service providers can be expressed by

$$\frac{\partial U_i^{\text{esp}}}{\partial p_i} < 0, \tag{39}$$

$$\frac{\partial^2 U_i^{\text{esp}}}{\partial {p_i}^2} = 0. \tag{40}$$

Since the second derivative is equal to zero, the first derivative is less than zero. Therefore, the function U_i^{esp} is decreasing. The function has a maximum point, and there is only one. The conclusion is established.

In stage 1, the income of edge service providers can be expressed by

$$U_i^{\text{esp}} = -\upsilon_c \sum_{i=1}^{\Gamma}\left(\text{pay}_i^{\text{esp}} \times B_i^o\right) - \upsilon_e\left(\sum_{i=1}^{\Gamma}\left(P_f^o \times \frac{B_{i,ne}^o}{F_i^o} \times \varphi_{ne} + P_i^o \times \frac{B_{i,ns}^o}{C_o^{ns}} \times \varphi_{ns}\right)\right) - \upsilon_t\left(\sum_{i=1}^{\Gamma}\left(\left(\frac{1}{F_i^o} + \frac{1}{\text{Cap}^{ne}}\right) \times B_{i,ne}^o \times \varphi_{ne} + \left(\frac{1}{C_o^{ns}} + \frac{1}{\text{Cap}^{ns}}\right) \times B_{i,ns}^o \times \varphi_{ns}\right)\right). \tag{41}$$

For the first derivative and second partial derivative of the function for the payment of the edge service provider, the solutions shown in equations (42) and (43) can be obtained:

$$\frac{\partial U_i^{\text{esp}}}{\partial \text{pay}_i^{\text{esp}}} < 0, \tag{42}$$

$$\frac{\partial^2 U_i^{\text{esp}}}{\partial \text{pay}_i^{\text{esp}2}} = 0. \tag{43}$$

Since the second derivative is equal to zero, the first derivative is less than zero. Therefore, the function U_i^{esp} is decreasing. The function has a maximum point, and there is only one. The conclusion can be established.□ □

In summary, the edge service provider's strategy set can maximize the benefits, and the optimal strategy is unique. According to Lemmas 1–3, Theorem 1 is proved.

Data Availability

The simulation parameter data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the National Nature Science Foundation of China under Grant 61572095 and Grant 61877007.

References

[1] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5031–5044, 2019.

[2] Y. Chen, Z. Li, B. Yang, K. Nai, and K. Li, "A Stackelberg game approach to multiple resources allocation and pricing in mobile edge computing," *Future Generation Computer Systems*, vol. 108, pp. 273–287, 2020.

[3] L. Pu, X. Chen, J. Xu, and X. Fu, "D2D fogging: an energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3887–3901, 2016.

[4] Y. Jie, C. Guo, K.-K. R. Choo, C. Z. Liu, and M. Li, "Game-theoretic resource allocation for fog-based industrial internet of things environment," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3041–3052, 2020.

[5] T. Başar and G. J. Olsder, *Dynamic Noncooperative Game Theory*, SIAM, 1998.

[6] H. V. Stackelberg, *Marktform Und Gleichgewicht*, Springer, University of California, 1934.

[7] Q. Fan and N. Ansari, "Application aware workload allocation for edge computing-based IoT," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2146–2153, 2018.

[8] X. Niu, S. Shao, C. Xin et al., "Workload allocation mechanism for minimum service delay in edge computing-based power Internet of Things," *IEEE Access*, vol. 7, pp. 83771–83784, 2019.

[9] Y. Hao, Y. Miao, L. Hu, M. S. Hossain, G. Muhammad, and S. U. Amin, "Smart-Edge-CoCaCo: AI-enabled smart edge with joint computation, caching, and communication in heterogeneous IoT," *IEEE Network*, vol. 33, no. 2, pp. 58–64, 2019.

[10] M. S. Parwez and D. B. Rawat, "Resource allocation in adaptive virtualized wireless networks with mobile edge computing," in *2018 IEEE International Conference on Communications (ICC 2018)*, Chengdu, China, 2018.

[11] Z. Zhang, J. Wu, L. Chen, G. Jiang, and S.-K. Lam, "Collaborative task offloading with computation result reusing for mobile edge computing," *The Computer Journal*, vol. 62, no. 10, pp. 1450–1462, 2019.

[12] M. Guo, L. Li, and Q. Guan, "Energy-efficient and delay-guaranteed workload allocation in IoT-edge-cloud computing systems," *IEEE Access*, vol. 7, pp. 78685–78697, 2019.

[13] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber–wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4514–4526, 2018.

[14] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing towards balanced delay and power consumption," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1171–1181, 2018.

[15] C. He, R. Wang, and Z. Tan, "Energy-aware collaborative computation offloading over mobile edge computation empowered fiber-wireless access networks," *IEEE Access*, vol. 8, pp. 24662–24674, 2020.

[16] F. Liu, Z. Huang, and L. Wang, "Energy-efficient collaborative task computation offloading in cloud-assisted edge computing for IoT sensors," *Sensors (Basel)*, vol. 19, no. 5, article 1105, 2019.

[17] J. Wu, Z. Cao, Y. Zhang, and X. Zhang, "Edge-cloud collaborative computation offloading model based on improved partical swarm optimization in MEC," in *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 959–962, Tianjin, China, 2019.

[18] F. Li, H. Yao, J. Du, C. Jiang, and Y. Qian, "Stackelberg game-based computation offloading in social and cognitive industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5444–5455, 2020.

[19] M. D. Hossain, L. N. Huynh, T. Sultana et al., "Collaborative task offloading for overloaded mobile edge computing in small-cell networks," in *2020 International Conference on Information Networking (ICOIN)*, pp. 717–722, Barcelona, Spain, 2020.

[20] J. Wang, W. Wu, Z. Liao, A. K. Sangaiah, and R. Simon Sherratt, "An energy-efficient off-loading scheme for low latency in collaborative edge computing," *IEEE Access*, vol. 7, pp. 149182–149190, 2019.

[21] R.-I. Ciobanu, C. Dobre, M. Balanescu, and G. Suciu, "Data and task offloading in collaborative mobile fog-based networks," *IEEE Access*, vol. 7, pp. 104405–104422, 2019.

[22] B. He, S. Bi, H. Xing, and X. Lin, "Collaborative computation offloading in wireless powered mobile-edge computing systems," in *2019 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–7, Waikoloa, HI, USA, 2019.

[23] Y. Kim, C. Song, H. Han, H. Jung, and S. Kang, "Collaborative task scheduling for IoT-assisted edge computing," *IEEE Access*, vol. 8, pp. 216593–216606, 2020.

[24] M. Ding, D. Lopez-Perez, H. Claussen, and M. A. Kaafar, "On the fundamental characteristics of ultra-dense small cell networks," *IEEE Network*, vol. 32, no. 3, pp. 92–100, 2018.

[25] I. Rec, "G. 107-the E model, a computational model for use in transmission planning," *International Telecommunication Union*, vol. 8, no. 20, 2003.

[26] P. Li, Y. Wang, W. Zhang, and Y. Huang, "QoE-oriented two-stage resource allocation in femtocell networks," in *2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall)*, pp. 1–5, Vancouver, BC, Canada, 2014.

[27] M. Zeleny, *Multiple Criteria Decision Making Kyoto 1975*, Springer, Berlin Heidelberg, New York, NY, USA, 2012.

[28] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," in *the 1st ACM symposium on Cloud computing*, pp. 39–50, Indianapolis, Indiana, USA, 2010.